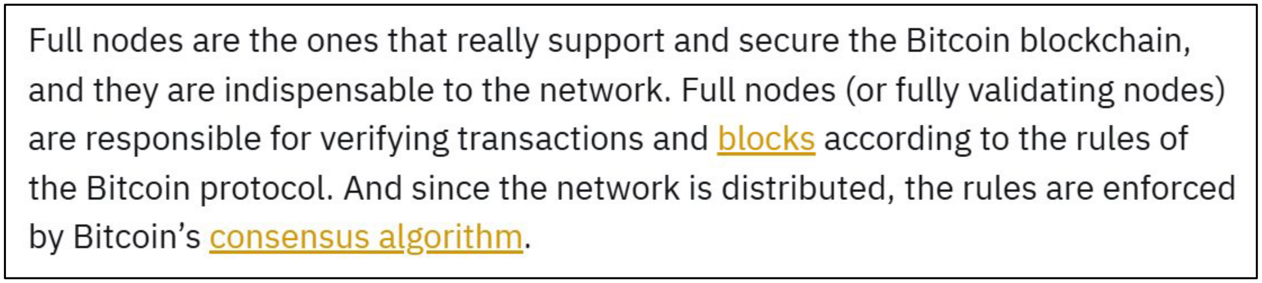
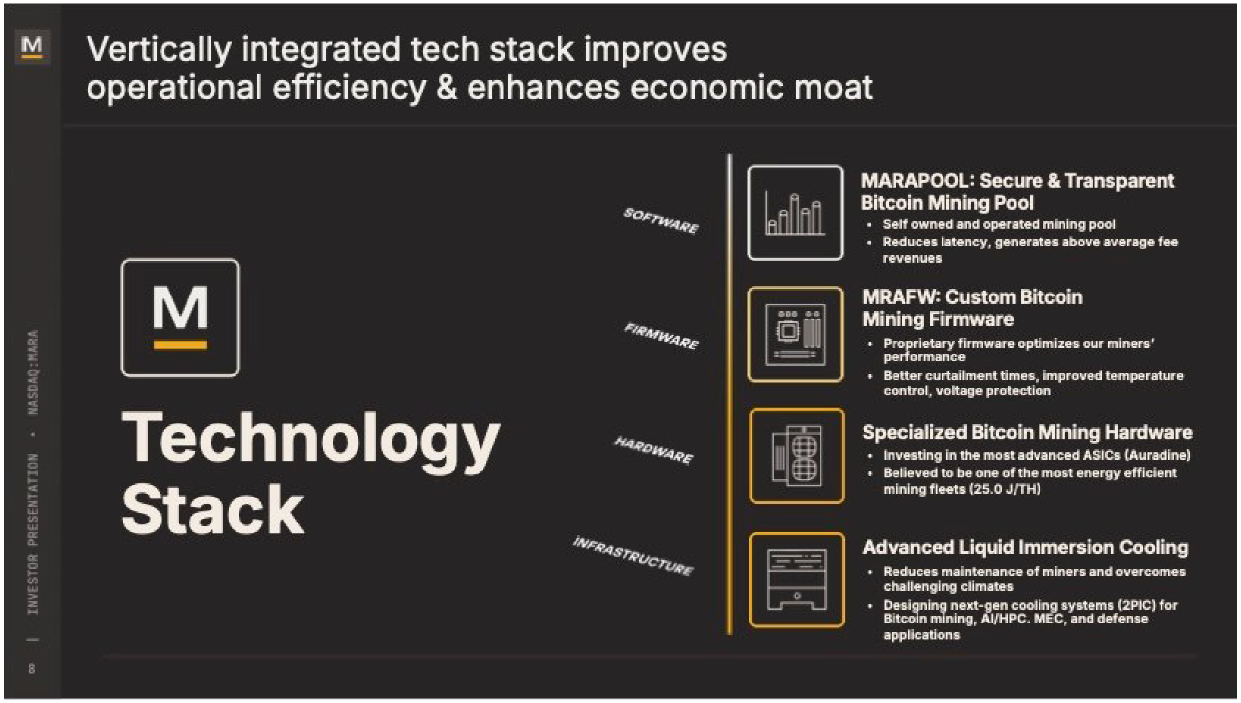
Exhibit 7

Exhibit 7: U.S. Patent No. 8,788,827

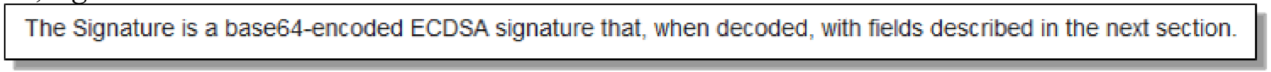
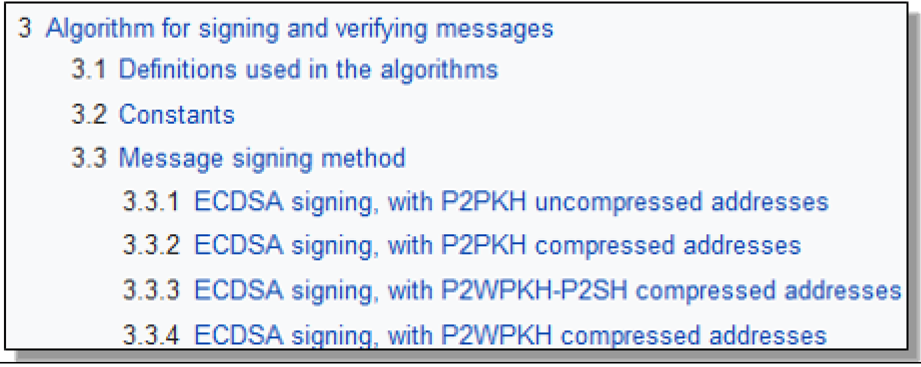
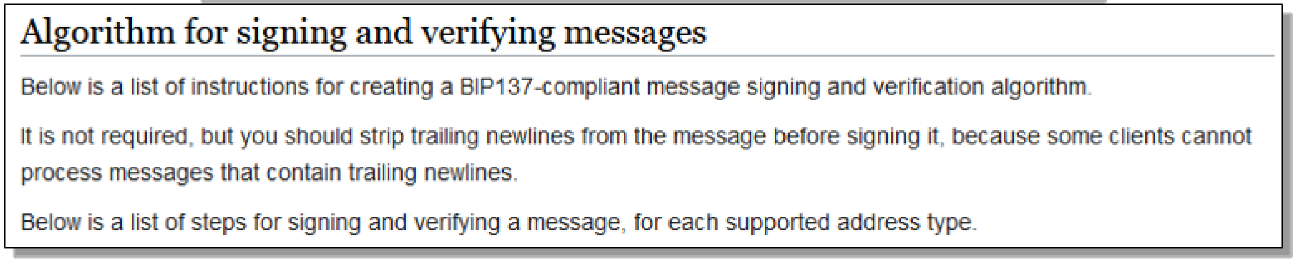
Claim 1	Exemplary Evidence of Infringement
1[pre] A computer-implemented method comprising:	MARA Holdings, Inc. (hereinafter "MARA") performs a computer-implemented method (e.g., verification of Bitcoin transactions). *See*, *e.g.*: "Marathon is a digital asset technology company that is principally engaged in producing or **'mining' digital assets with a focus on the Bitcoin ecosystem** … **The term 'Bitcoin' with a capital 'B' is used to denote the Bitcoin protocol** which implements a highly available, public, permanent, and decentralized ledger." (Emphasis added) *See*, *e.g.*, MARA Holdings, Inc., Annual report pursuant to Section 13 and 15(d), (Form 10-K/A), at F-9, filed May 24, 2024, available at https://ir.mara.com/sec-filings/all-sec-filings/content/0001628280-24-025261/mara-20231231.htm. "As Operator, the Company **provides transaction verification services** to the transaction requester, in addition to the Bitcoin network. **Transaction verification services are an output of the Company's ordinary activities**; therefore, the Company views the transaction requester as a customer and **recognizes the transaction fees as revenue from contracts with customers under ASC 606**. The Bitcoin network is not an entity such that it may not meet the definition of a customer; however, the Company has concluded that **it is appropriate to apply ASC 606 by analogy to block rewards earned from the Bitcoin network**."(Emphasis added). *See*, *e.g.*, MARA Holdings., Inc., Quarterly report, (Form 10-Q), at Note 4 – Revenues, filed November 12, 2024, available at https://www.sec.gov/ix?doc=/Archives/edgar/data/0001507605/000162828024047148/mara-20240930.htm. "The Bitcoin protocol is the technology that enables Bitcoin to function as a decentralized, peer-to-peer payment network. This open-source software, which sets the rules and processes that govern the Bitcoin network, is maintained and improved by a community of developers around the world known as Bitcoin Core developers … 'At Marathon, we have historically focused on supporting Bitcoin by adding hash rate, which helps secure the network, and now, we are supporting those who

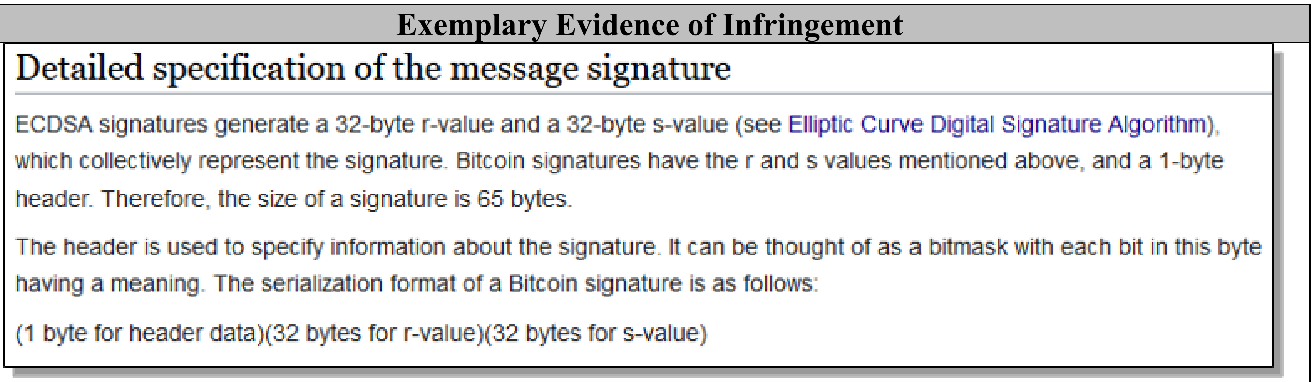
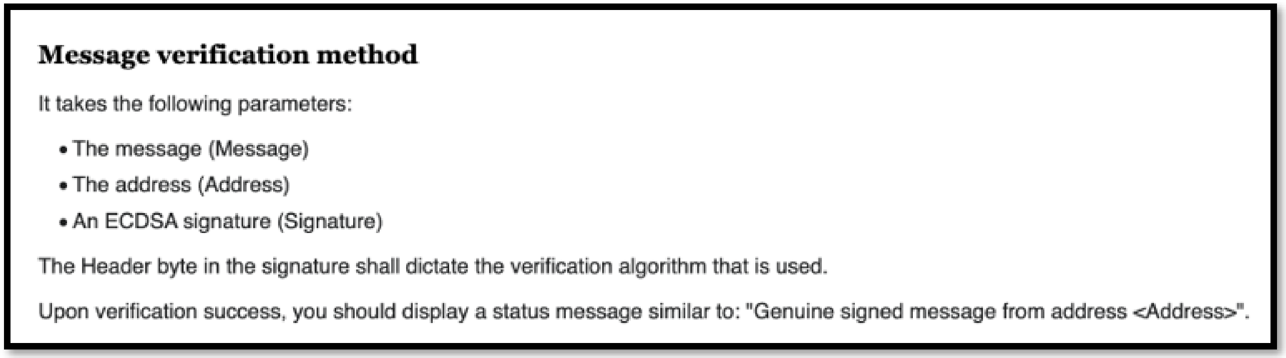
Claim 1	Exemplary Evidence of Infringement
	maintain **the open-source protocol on which we all depend** by contributing to Brink,' said Fred Thiel, Marathon's chairman and CEO." (Emphasis added) *See*, *e.g.*, Marathon Holdings Collaborates with Brink To Raise Up to $1 Million To Support Bitcoin Core Developers, GlobeNewswire (May 18, 2023), available at https://www.globenewswire.com/news-release/2023/05/18/2672276/0/en/Marathon-Digital-Holdings-Collaborates-with-Brink-To-Raise-Up-to-1-Million-To-Support-Bitcoin-Core-Developers.html. "**Bitcoin signed messages have three parts, which are the Message, Address, and Signature**. The message is the actual message text - all kinds of text is supported, but it is recommended to avoid using non-ASCII characters in the signature because they might be encoded in different character sets, preventing signature verification from succeeding. The address is a legacy, nested segwit, or native segwit address. Message signing from legacy addresses was added by Satoshi himself and therefore does not have a BIP. **Message signing from segwit addresses has been added by BIP137 … The Signature is a base64-encoded ECDSA signature** that, when decoded, with fields described in the next section." (Emphasis added) *See*, *e.g.,* Message Signing, https://en.bitcoin.it/wiki/Message_signing. "This document describes a signature format for **signing messages with Bitcoin private keys**. The specification is intended to describe the standard for signatures of messages that can be signed and verified between different clients that exist in the field today." (Emphasis added) *See*, *e.g.*, Bitcoin BIP137, https://github.com/bitcoin/bips/blob/master/bip-0137.mediawiki. For example, MARA utilizes a computer (*e.g.*, a node or miner in a peer-to-peer network) (*e.g.*, using ASICs) when verifying transactions under the Bitcoin protocol. *See*, *e.g.*:

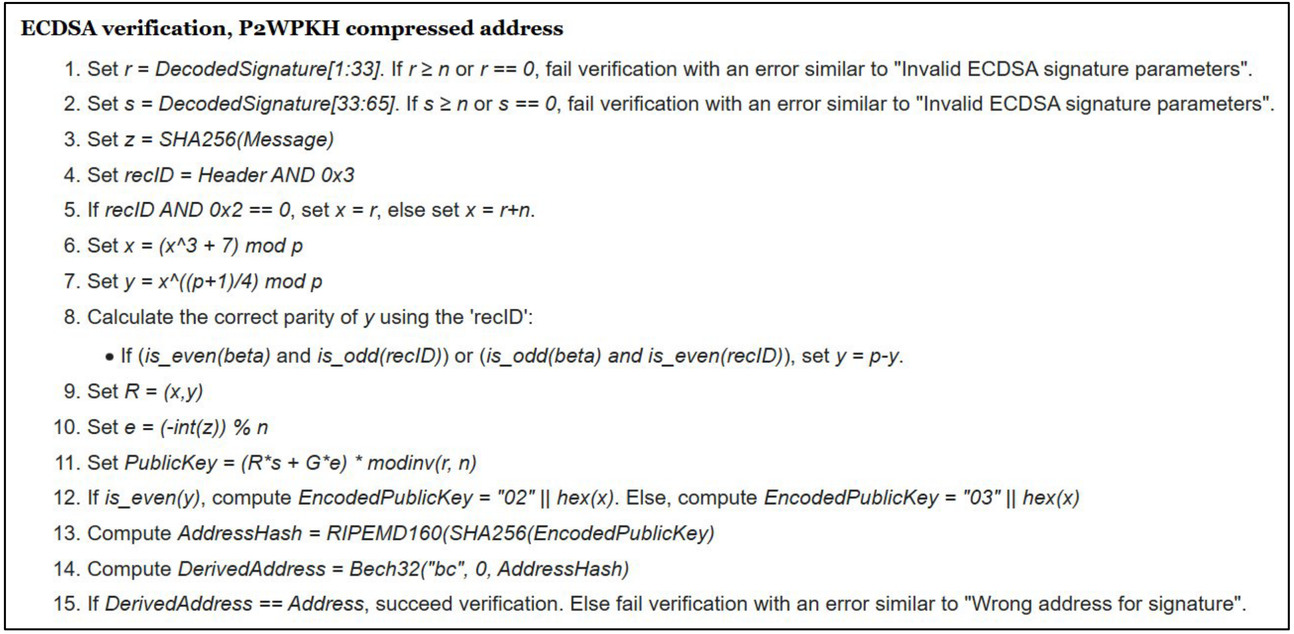
Claim 1	Exemplary Evidence of Infringement
	“Bitcoin is a decentralized digital currency that enables instant payments to anyone, anywhere in the world. Bitcoin uses peer-to-peer technology to operate with no central authority: transaction management and money issuance are carried out collectively by the network.” *See*, *e.g.*, Welcome to the Bitcoin Wiki, https://en.bitcoin.it/wiki/Main_Page.  Full nodes are the ones that really support and secure the Bitcoin blockchain, and they are indispensable to the network. Full nodes (or fully validating nodes) are responsible for verifying transactions and blocks according to the rules of the Bitcoin protocol. And since the network is distributed, the rules are enforced by Bitcoin’s consensus algorithm. *See*, *e.g.*, Node, https://academy.binance.com/en/glossary/node. In the world of cryptocurrencies, the term ASIC is widely used to refer to the specialized hardware that are being developed and regularly improved by companies such as Bitmain and Halong Mining. These hardware are designed with the sole intention of mining Bitcoin (or other cryptocurrencies). There are some coins that cannot be effectively mined using ASIC miners and, as such, may be referred to as ASIC-resistant cryptocurrencies. *See*, *e.g.*, Application-Specific Integrated Circuit (ASIC), https://academy.binance.com/en/glossary/application-specific-integrated-circuit.

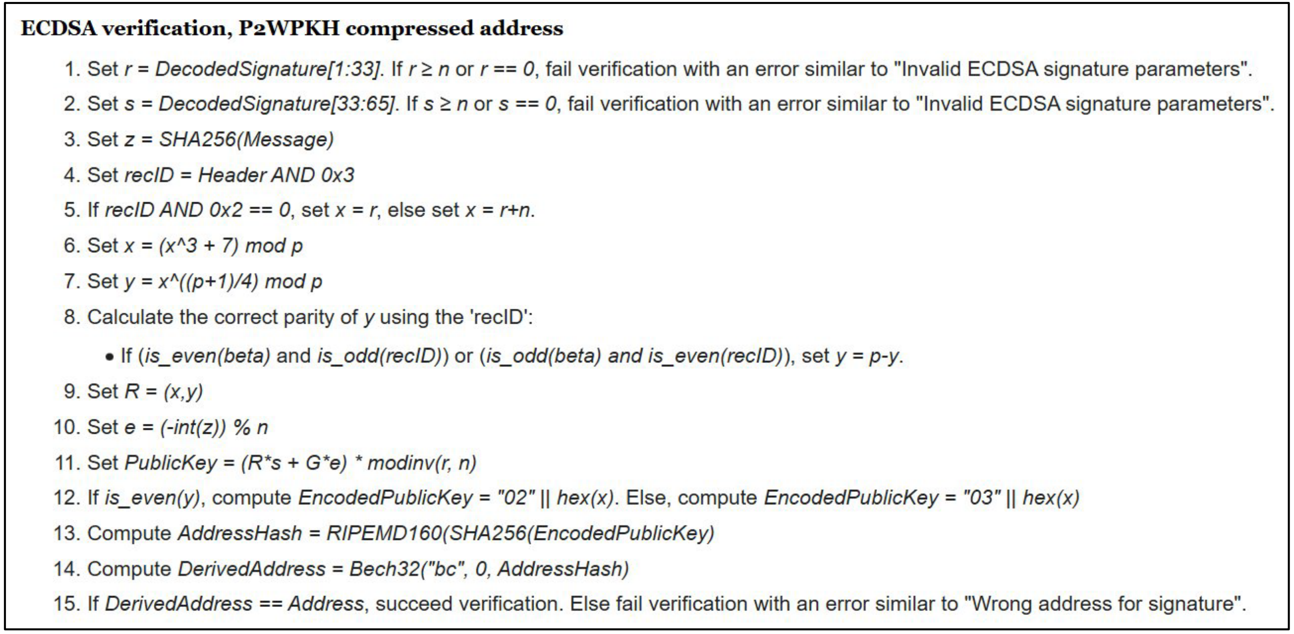
Claim 1	Exemplary Evidence of Infringement
	 SOFTWARE FIRMWARE HARDWARE INFRASTRUCTURE INVESTOR PRESENTATION • NASDAQ:MARA | 8 Vertically integrated tech stack improves operational efficiency & enhances economic moat **Technology Stack** **MARAPOOL: Secure & Transparent Bitcoin Mining Pool** • Self owned and operated mining pool • Reduces latency, generates above average fee revenues **MRAFW: Custom Bitcoin Mining Firmware** • Proprietary firmware optimizes our miners' performance • Better curtailment times, improved temperature control, voltage protection **Specialized Bitcoin Mining Hardware** • Investing in the most advanced ASICs (Auradine) • Believed to be one of the most energy efficient mining fleets (25.0 J/TH) **Advanced Liquid Immersion Cooling** • Reduces maintenance of miners and overcomes challenging climates • Designing next-gen cooling systems (2PIC) for Bitcoin mining, AI/HPC. MEC, and defense applications *See*, *e.g.*, https://d1io3yog0oux5.cloudfront.net/_2dc31b1794f998d5238d31da962c40f6/marathondh/db/417/7503/pdf/24Q3-INVESTOR+PRESENTATION.pdf. MARA performs the method using a computer. *See*, *e.g.*: "Our core business is bitcoin mining, and we produce, or 'mine,' bitcoin using one of the industry's largest and most energy-efficient fleets of **specialized computers** while providing dispatchable compute as an optionality to the electric grid operators to balance electric demands on the grid." (Emphasis added)

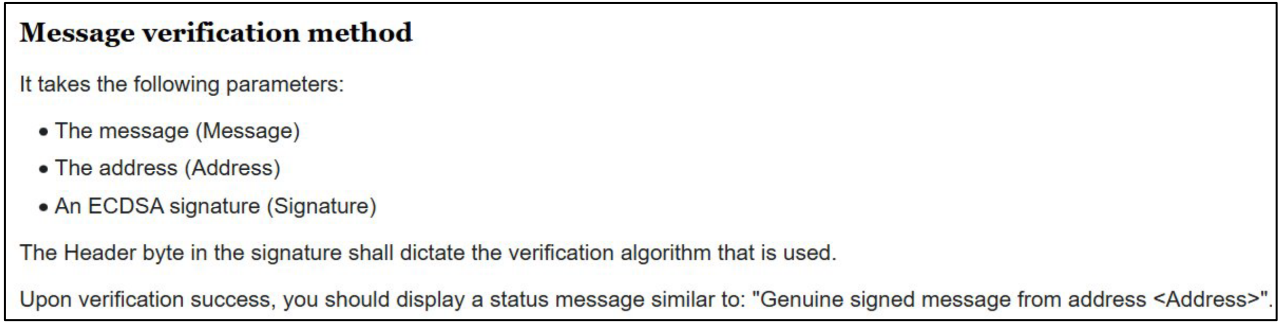
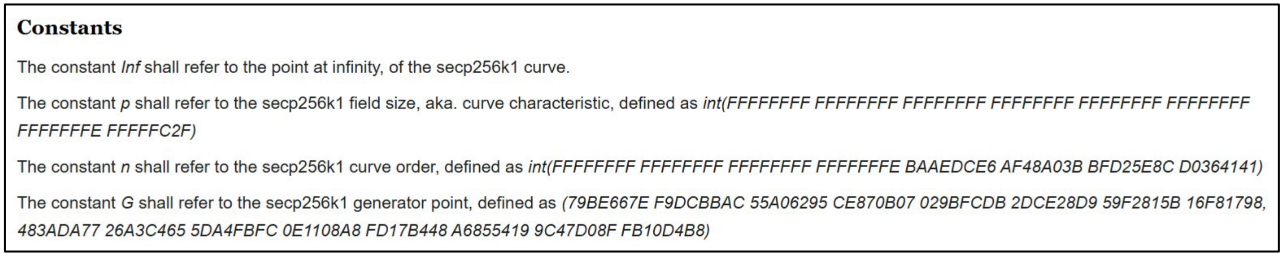
Claim 1	Exemplary Evidence of Infringement
	See, *e.g.*, MARA Holdings, Inc., Form 10-K/A, at 6, filed March 3, 2025, available at https://ir.mara.com/sec-filings/all-sec-filings/content/0001628280-24-025261/mara-20231231.htm. "Over the past three years, digital asset mining operations have evolved from individual users mining with **computer processors, graphics processing units and first-generation mining rigs**. New processing power brought onto the digital asset networks is predominantly added by professionalized mining operations, which may use **proprietary hardware or sophisticated machines**." (Emphasis added) *See*, *e.g.*, MARA Holdings, Inc., Form 10-K/A, at 21, filed March 3, 2025, available at https://ir.mara.com/sec-filings/all-sec-filings/content/0001628280-24-025261/mara-20231231.htm. "As of December 31, 2024, we operated approximately 400,000 bitcoin mining **ASICs**, capable of producing 53.2 EH/s with an efficiency of 19.2 joules per terahash, which is among the most efficient in the industry." (Emphasis added) *See*, *e.g.*, MARA Holdings, Inc., Form 10-K/A, at 21, filed March 3, 2025, available at https://ir.mara.com/sec-filings/all-sec-filings/content/0001628280-24-025261/mara-20231231.htm. "Miners, which operate **specialized hardware, known as bitcoin mining rigs or application-specific integrated circuits ("ASICs")**, then compete to process these unconfirmed transactions into a 'block.'" (Emphasis added) *See*, *e.g.*, MARA Holdings, Inc., Form 10-K/A, at 6, filed March 3, 2025, available at https://ir.mara.com/sec-filings/all-sec-filings/content/0001628280-24-025261/mara-20231231.htm.

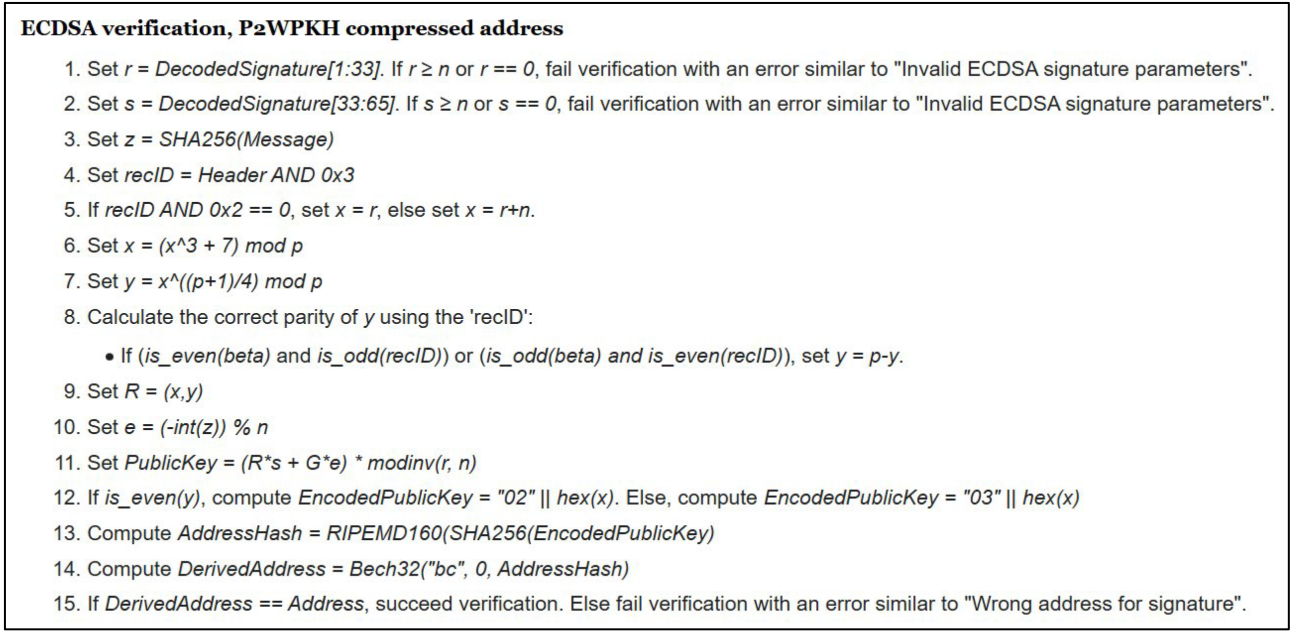
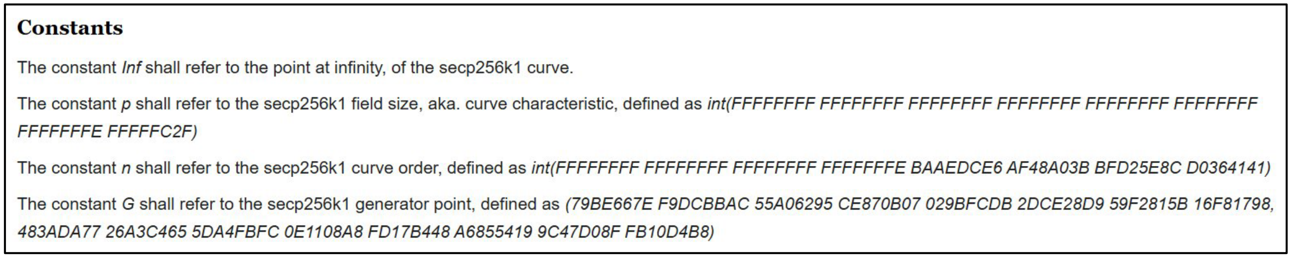
Claim 1	Exemplary Evidence of Infringement
1[a] receiving, from a signer, a signature on a message M, wherein the signature includes a first signature component r and a second signature component s;	MARA's miners receive, from a signer (*e.g.*, Bitcoin transferor), a signature on a message M, wherein the signature includes a first signature component r and a second signature component s. *See*, *e.g.*:  The Signature is a base64-encoded ECDSA signature that, when decoded, with fields described in the next section.  3 Algorithm for signing and verifying messages 3.1 Definitions used in the algorithms 3.2 Constants 3.3 Message signing method 3.3.1 ECDSA signing, with P2PKH uncompressed addresses 3.3.2 ECDSA signing, with P2PKH compressed addresses 3.3.3 ECDSA signing, with P2WPKH-P2SH compressed addresses 3.3.4 ECDSA signing, with P2WPKH compressed addresses  Algorithm for signing and verifying messages Below is a list of instructions for creating a BIP137-compliant message signing and verification algorithm. It is not required, but you should strip trailing newlines from the message before signing it, because some clients cannot process messages that contain trailing newlines. Below is a list of steps for signing and verifying a message, for each supported address type. *See*, *e.g.*, Message signing, https://en.bitcoin.it/wiki/Message_signing. For example, MARA's miners receive, from a signer, a signature (*e.g.*, ECDSA signature) on a message M, wherein the signature includes a first signature component r (*e.g.*, the r-value) and a second signature component s (*e.g.*, the s-value). *See*, *e.g.*:

Claim 1	Exemplary Evidence of Infringement
	 **Detailed specification of the message signature** ECDSA signatures generate a 32-byte r-value and a 32-byte s-value (see Elliptic Curve Digital Signature Algorithm), which collectively represent the signature. Bitcoin signatures have the r and s values mentioned above, and a 1-byte header. Therefore, the size of a signature is 65 bytes. The header is used to specify information about the signature. It can be thought of as a bitmask with each bit in this byte having a meaning. The serialization format of a Bitcoin signature is as follows: (1 byte for header data)(32 bytes for r-value)(32 bytes for s-value)  **Message verification method** It takes the following parameters: • The message (Message) • The address (Address) • An ECDSA signature (Signature) The Header byte in the signature shall dictate the verification algorithm that is used. Upon verification success, you should display a status message similar to: "Genuine signed message from address <Address>". *See*, *e.g.*, Message signing, https://en.bitcoin.it/wiki/Message_signing.
1[b] obtaining an elliptic curve point associated with the first signature component r; and	MARA's miners obtain an elliptic curve point (*e.g.*, R = (x,y)) associated with the first signature component r (*e.g.*, x is associated with r, and y is associated with x). *See*, *e.g.*:

Claim 1	Exemplary Evidence of Infringement
	 **ECDSA verification, P2WPKH compressed address** 1. Set r = DecodedSignature[1:33]. If r ≥ n or r == 0, fail verification with an error similar to "Invalid ECDSA signature parameters". 2. Set s = DecodedSignature[33:65]. If s ≥ n or s == 0, fail verification with an error similar to "Invalid ECDSA signature parameters". 3. Set z = SHA256(Message) 4. Set recID = Header AND 0x3 5. If recID AND 0x2 == 0, set x = r, else set x = r+n. 6. Set x = (x^3 + 7) mod p 7. Set y = x^((p+1)/4) mod p 8. Calculate the correct parity of y using the 'recID': • If (is_even(beta) and is_odd(recID)) or (is_odd(beta) and is_even(recID)), set y = p-y. 9. Set R = (x,y) 10. Set e = (-int(z)) % n 11. Set PublicKey = (R*s + G*e) * modinv(r, n) 12. If is_even(y), compute EncodedPublicKey = "02" || hex(x). Else, compute EncodedPublicKey = "03" || hex(x) 13. Compute AddressHash = RIPEMD160(SHA256(EncodedPublicKey) 14. Compute DerivedAddress = Bech32("bc", 0, AddressHash) 15. If DerivedAddress == Address, succeed verification. Else fail verification with an error similar to "Wrong address for signature". *See*, *e.g.*, Message signing, https://en.bitcoin.it/wiki/Message_signing.
1[c] generating, by operation of a cryptographic module comprising one or more processors, a public key of the signer based on the elliptic curve point and a hash value e computed from the message M;	MARA's miners generate, by operation of a cryptographic module (*e.g.*, a node in a peer-to-peer network) comprising one or more processors (*e.g.*, ASIC, GPU, etc.), a public key (*e.g.*, PublicKey) of the signer based on the elliptic curve point (*e.g.*, R) and a hash value e (e.g., e) computed from the message M. *See*, *e.g.*:

Claim 1	Exemplary Evidence of Infringement
	 **ECDSA verification, P2WPKH compressed address** 1. Set *r = DecodedSignature[1:33]*. If *r ≥ n* or *r == 0*, fail verification with an error similar to "Invalid ECDSA signature parameters". 2. Set *s = DecodedSignature[33:65]*. If *s ≥ n* or *s == 0*, fail verification with an error similar to "Invalid ECDSA signature parameters". 3. Set *z = SHA256(Message)* 4. Set *recID = Header AND 0x3* 5. If *recID AND 0x2 == 0*, set *x = r*, else set *x = r+n*. 6. Set *x = (x^3 + 7) mod p* 7. Set *y = x^((p+1)/4) mod p* 8. Calculate the correct parity of *y* using the 'recID': • If (*is_even(beta)* and *is_odd(recID)*) or (*is_odd(beta) and is_even(recID)*), set *y = p-y*. 9. Set *R = (x,y)* 10. Set *e = (-int(z)) % n* 11. Set *PublicKey = (R*s + G*e) * modinv(r, n)* 12. If *is_even(y)*, compute *EncodedPublicKey = "02" || hex(x)*. Else, compute *EncodedPublicKey = "03" || hex(x)* 13. Compute *AddressHash = RIPEMD160(SHA256(EncodedPublicKey)* 14. Compute *DerivedAddress = Bech32("bc", 0, AddressHash)* 15. If *DerivedAddress == Address*, succeed verification. Else fail verification with an error similar to "Wrong address for signature". *See*, *e.g.*, Message signing, https://en.bitcoin.it/wiki/Message_signing. For example, the hash value e (*e.g.*, e) is computed from the message M using the formula e = (-int(z)) % n, where z is the hash value of the message (*e.g.*, SHA256) and n refers to the secp256k1 curve order, as shown below. *See*, *e.g.*:

Claim 1	Exemplary Evidence of Infringement
	 **Message verification method** It takes the following parameters: - The message (Message) - The address (Address) - An ECDSA signature (Signature) The Header byte in the signature shall dictate the verification algorithm that is used. Upon verification success, you should display a status message similar to: "Genuine signed message from address <Address>".  **Constants** The constant *Inf* shall refer to the point at infinity, of the secp256k1 curve. The constant *p* shall refer to the secp256k1 field size, aka. curve characteristic, defined as *int(FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F)* The constant *n* shall refer to the secp256k1 curve order, defined as *int(FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141)* The constant *G* shall refer to the secp256k1 generator point, defined as *(79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798, 483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8)* *See*, *e.g.*, Message signing, https://en.bitcoin.it/wiki/Message_signing.
1[d] wherein the elliptic curve point comprises a first elliptic curve point R, the public key of the signer comprises a second elliptic curve point Q, generating the public key of the signer comprises computing Q=r−1 (sR−eG), and G comprises a generator of an elliptic curve group that includes the first elliptic curve point R and the second elliptic curve point Q.	The elliptic curve point comprises a first elliptic curve point R, the public key of the signer comprises a second elliptic curve point Q, and generating the public key of the signer comprises computing Q=r−1 (sR−eG), and G comprises a generator of an elliptic curve group that includes the first elliptic curve point R and the second elliptic curve point Q. *See*, *e.g.*:

Claim 1	Exemplary Evidence of Infringement
	 **ECDSA verification, P2WPKH compressed address** 1. Set r = DecodedSignature[1:33]. If r ≥ n or r == 0, fail verification with an error similar to "Invalid ECDSA signature parameters". 2. Set s = DecodedSignature[33:65]. If s ≥ n or s == 0, fail verification with an error similar to "Invalid ECDSA signature parameters". 3. Set z = SHA256(Message) 4. Set recID = Header AND 0x3 5. If recID AND 0x2 == 0, set x = r, else set x = r+n. 6. Set x = (x^3 + 7) mod p 7. Set y = x^((p+1)/4) mod p 8. Calculate the correct parity of y using the 'recID': • If (is_even(beta) and is_odd(recID)) or (is_odd(beta) and is_even(recID)), set y = p-y. 9. Set R = (x,y) 10. Set e = (-int(z)) % n 11. Set PublicKey = (R*s + G*e) * modinv(r, n) 12. If is_even(y), compute EncodedPublicKey = "02" || hex(x). Else, compute EncodedPublicKey = "03" || hex(x) 13. Compute AddressHash = RIPEMD160(SHA256(EncodedPublicKey) 14. Compute DerivedAddress = Bech32("bc", 0, AddressHash) 15. If DerivedAddress == Address, succeed verification. Else fail verification with an error similar to "Wrong address for signature". *See*, *e.g.*, Message signing, https://en.bitcoin.it/wiki/Message_signing. For example, the equation Q=r−1 (sR−eG) is used to determine Q, which is the PublicKey (a point on the elliptic curve secp256k1). *See*, *e.g.*:  **Constants** The constant Inf shall refer to the point at infinity, of the secp256k1 curve. The constant p shall refer to the secp256k1 field size, aka. curve characteristic, defined as int(FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F) The constant n shall refer to the secp256k1 curve order, defined as int(FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141) The constant G shall refer to the secp256k1 generator point, defined as (79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798, 483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8) *See*, *e.g.*, Message signing, https://en.bitcoin.it/wiki/Message_signing.